HoGent
BEDRIJF
EN
ORGANISATIE

MODEL-VIEW-CONTROLLER: DEEL 1 Observer Pattern

1. INLEIDING

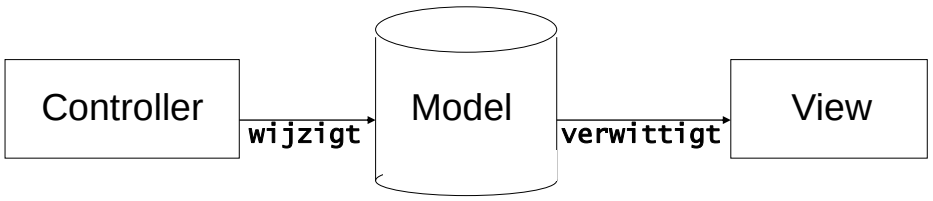
- **Wikipedia:**

Model-View-Controller (of MVC) is een structuur (of patroon) dat het ontwerp van complexe toepassingen opdeelt in drie eenheden met verschillende verantwoordelijkheden: Datamodel (**Model**), Datapresentatie (**View**) en Applicatielogica (**Controller**).

Het scheiden van deze verantwoordelijkheden **bevordert de leesbaarheid** en **herbruikbaarheid** van code. Het maakt ook dat bijvoorbeeld **veranderingen in de interface niet direct invloed hebben op het datamodel en vice versa**.

MVC werd voor het eerst gebruikt in de eerste implementaties van Smalltalk.

2. Model-View-Controller architectuur

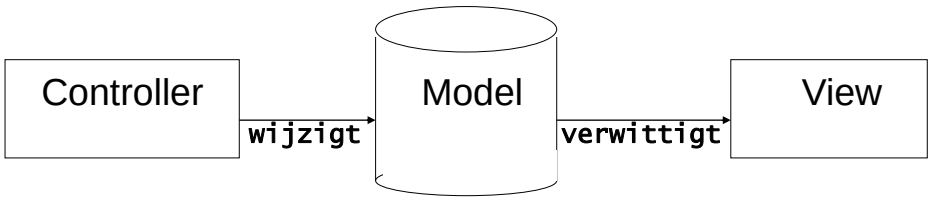


Controller: invoerverwerking
Model: de data
View: presentatie van de data

HoGent

3

2. Model-View-Controller architectuur



Controller wijzigt Model: Bij gebruikersinvoer zal de controller het model wijzigen met de zojuist ingegeven data.

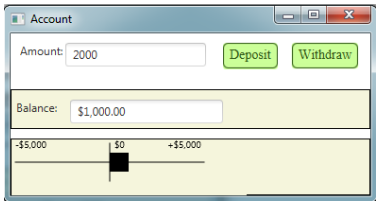
Model verwittigt View: wanneer het model is gewijzigd, za model de view verwittigen, zodat de view zijn grafische voorstelling kan aanpassen.

4

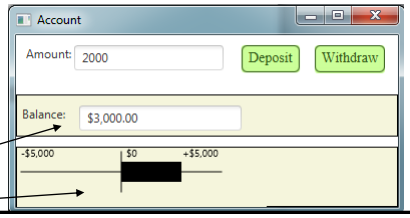
3. Observer-patroon

Het observer-patroon heeft een Model-View-Controller architectuur.

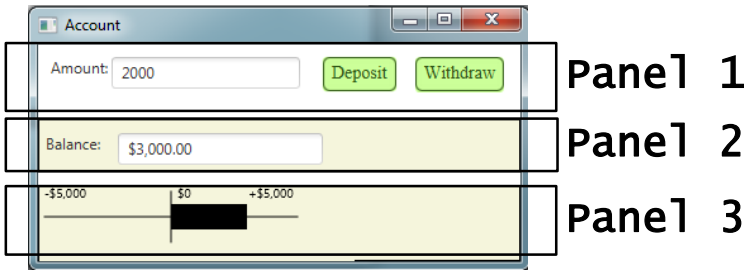
Voorbeeld: De gebruiker kan geld afhalen of geld storten op zijn rekening. De gebruiker geeft 2000 in



en drukt op “Deposit”:



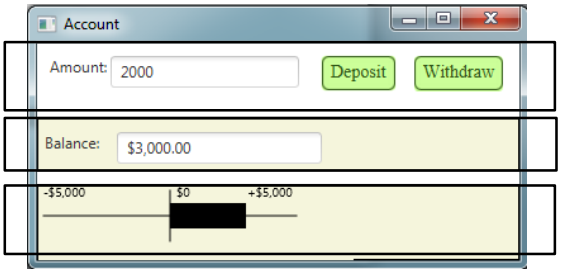
Frame bestaat uit drie panels:



Indien de gebruiker in panel1 geld stort of afhaalt, dan dienen panel2 en panel3 hun grafische voorstelling van het saldo aan te passen.

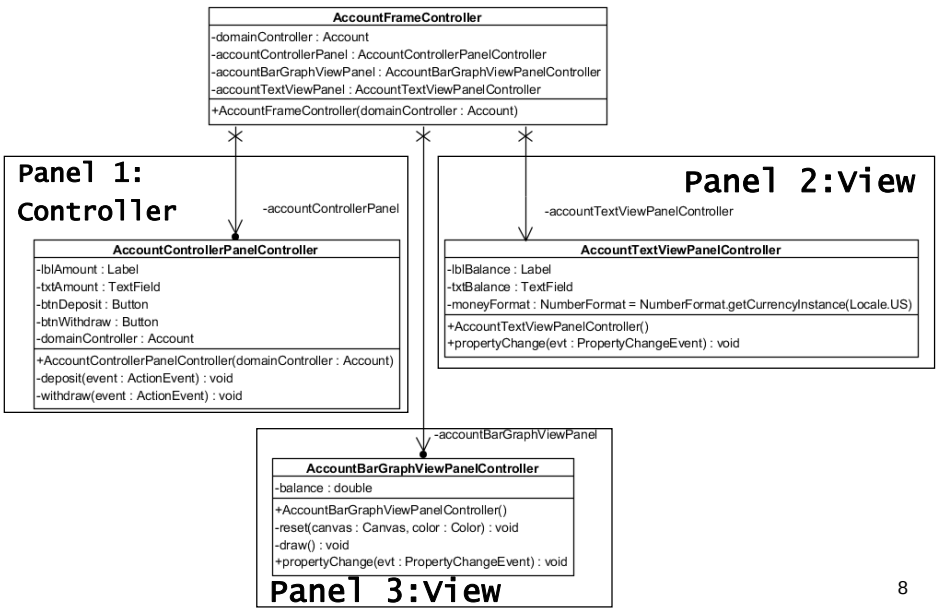
Panel1, Panel2 en Panel3 zijn drie afzonderlijke klassen. Ze hebben geen link naar elkaar.

Voordeel: **herbruikbaarheid**.



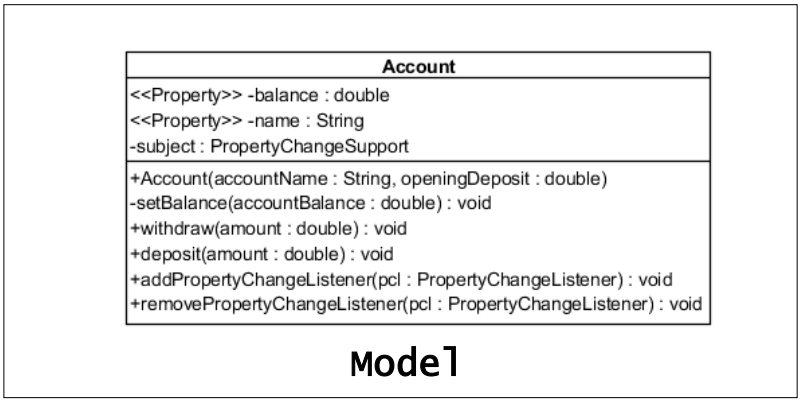
- Panel1 is de controller
- Panel2 en Panel3 zijn de views. In Java gebruiken we hiervoor de interface PropertyChangeListener.
- De domeinklasse Account (bevat het saldo) is het model. In Java gebruiken we hiervoor de klasse PropertyChangeSupport.

package gui:



8

package domein:



9

public class AccountFrameController extends GridPane {

Frame

private final Account domainController;
private final AccountControllerPanelController accountControllerPanel;
private final AccountTextViewPanelController accountTextViewPanel;
private final AccountBarGraphViewPanelController
accountBarGraphViewPanel;

DomeinController

public AccountFrameController(Account domainController) {

this.domainController = domainController;
accountControllerPanel = new
AccountControllerPanelController(domainController);

HoGent

Frame

```
public AccountFrameController(Account domainController) {
        …
        accountTextViewPanel =
                    new AccountTextViewPanelController();
        …
        domainController.addPropertyChangeListener(
                                        accountTextViewPanel);
```

View panel2

```
public AccountTextViewPanelController() {
        …
```

HoGent

Frame

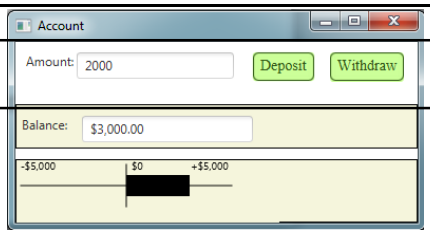
```
public AccountFrameController(Account domainController) {
        …
        accountBarGraphViewPanel =
                new AccountBarGraphViewPanelController();
        …
        domainController.addPropertyChangeListener(
                                    accountBarGraphViewPanel);
```

View panel3

```
public AccountBarGraphViewPanelController() {
        …
```

HoGent

Panel 1




```
import domain.Account;
…
public class AccountControllerPanelController extends HBox {
   @FXML
   private TextField txtAmount;
   @FXML
   private Button btnDeposit;
   @FXML
   private Button btnWithdraw;

  private final Account domainController;

  public AccountControllerPanelController(Account domainController) {
      this.domainController = domainController;
  }
```

Controller

HoGent

Controller

Deposit

```
@FXML
private void deposit(ActionEvent event)
{
      try {
          domainController.deposit(
                  Double.parseDouble(txtAmount.getText()));
      }
      catch (NumberFormatException exception) {
         new Alert(AlertType.ERROR,
                  "Please enter a valid    amount").showAndWait();
      }
      catch (IllegalArgumentException exception) {
          new Alert(AlertType.ERROR,
                  exception.getMessage()).showAndWait();
      }
}
```

Controller

Withdraw

```
@FXML
private void withdraw(ActionEvent event) {
   try {
      domainController.withdraw(
            Double.parseDouble(txtAmount.getText()));
   }
   catch (NumberFormatException exception) {
      new Alert(AlertType.ERROR,
            "Please enter a valid    amount").showAndWait();
   }
   catch (IllegalArgumentException exception) {
       new Alert(AlertType.ERROR,
            exception.getMessage()).showAndWait();
   }
}
```

Model

```
package domein;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;

public class Account
{
   private double balance;
   private PropertyChangeSupport subject;
   …

   public Account(String accountName, double openingDeposit) {
     name = accountName;
     subject = new PropertyChangeSupport(this);
     setBalance(openingDeposit);
   }
```

HoGent

Model

```
// bedrag storten
public void deposit(double amount) throws IllegalArgumentException
{
 if (amount < 0)
      throw new IllegalArgumentException(
                    "Cannot deposit negative amount");

 // update Account balance
 setBalance(getBalance() + amount);
}
```

HoGent

Model

```
// bedrag afhalen
public void withdraw(double amount) throws
IllegalArgumentException
{
 if (amount < 0)
     throw new IllegalArgumentException(
         "Cannot withdraw negative amount");

 // update Account balance
 setBalance(getBalance() - amount);
}

public double getBalance()
{
 return balance;
}
```

HoGent

Model

```
// bedrag wijzigen en alle observers van de
// verandering op de hoogte stellen
private void setBalance(double accountBalance)
{

  /* alle Observers verwittigen dat het model
     gewijzigd is, in ons voorbeeld worden de views
     accountTextViewPanel  en
     accountBarGraphViewPanel verwittigd */
    //voor elke observer wordt de methode propertyChange
    //opgeroepen
   subject.firePropertyChange(
         "balance", this.balance, accountBalance);
                    //oude waarde   nieuwe waarde

  balance = accountBalance;
}
```

Model

```
public void addPropertyChangeListener(PropertyChangeListener pcl) {
      subject.addPropertyChangeListener(pcl);
      pcl.propertyChange(
       new PropertyChangeEvent(pcl, "balance", null, balance));
  }

  public void removePropertyChangeListener(PropertyChangeListener pcl)
  {
      subject.removePropertyChangeListener(pcl);
  }

}//einde klasse
```

HoGent

panel 2

```
public class AccountTextViewPanelController
   extends HBox  implements PropertyChangeListener {

  @FXML
  private TextField txtBalance;

  // NumberFormat for US dollars
  private NumberFormat moneyFormat =
    NumberFormat.getCurrencyInstance(Locale.US);

  public AccountTextViewPanelController() {…}

  @Override
  public void propertyChange(PropertyChangeEvent evt) {
      double balance = (Double) evt.getNewValue();
      txtBalance.setText(moneyFormat.format(balance));
  }}
```

View panel 3

```
public class AccountBarGraphViewPanelController extends
   Pane  implements PropertyChangeListener {

  private double balance;

  public AccountBarGraphViewPanelController() {
     FXMLLoader loader = new FXMLLoader(getClass().getResource(
              "AccountBarGraphViewPanel.fxml"));
     loader.setRoot(this);
     loader.setController(this);
     try {
        loader.load();
     } catch (IOException ex) {
        throw new RuntimeException(ex);
     }
  }
```

HoGent

```
@Override
public void propertyChange(PropertyChangeEvent evt) {
    this.balance = (Double) evt.getNewValue();
    draw();
}

private void reset(Canvas canvas, Color color) {
    GraphicsContext gc =
 canvas.getGraphicsContext2D();
    gc.setFill(color);
    gc.fillRect(1, 1, canvas.getWidth(), canvas.getHeight());
}
```

View panel 3

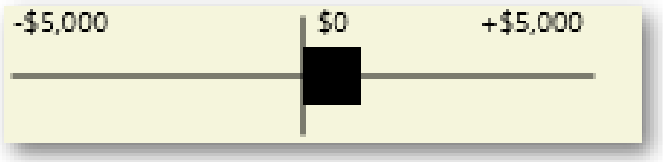
HoGent

```
private void draw() {
    Group root = new Group();
    Scene s = new Scene(root, 300, 300, Color.BLACK);
    final Canvas canvas = new Canvas(250, 250);
    GraphicsContext gc =
            canvas.getGraphicsContext2D();

    reset(canvas, Color.BEIGE);

    int barLength = (int) ((balance / 10000.0) * 200);
    if (balance >= 0.0) {
        gc.setFill(Color.BLACK);
        gc.fillRect(105, 15, barLength, 20);
    } // if balance is negative, draw graph in red
    else {
        gc.setFill(Color.RED);
        gc.fillRect(105 + barLength, 15, -barLength, 20);
    }
```

View panel 3

```
    // draw vertical and horizontal axes
    gc.setFill(Color.BLACK);
    gc.strokeLine(5, 25, 205, 25);
    gc.strokeLine(105, 5, 105, 45);

    // draw graph labels
    gc.setFont(new Font("SansSerif", 10));
    gc.fillText("-$5,000", 5, 10);
    gc.fillText("$0", 110, 10);
    gc.fillText("+$5,000", 166, 10);
    getChildren().add(canvas);
  }
}
```

View panel 3



HoGent

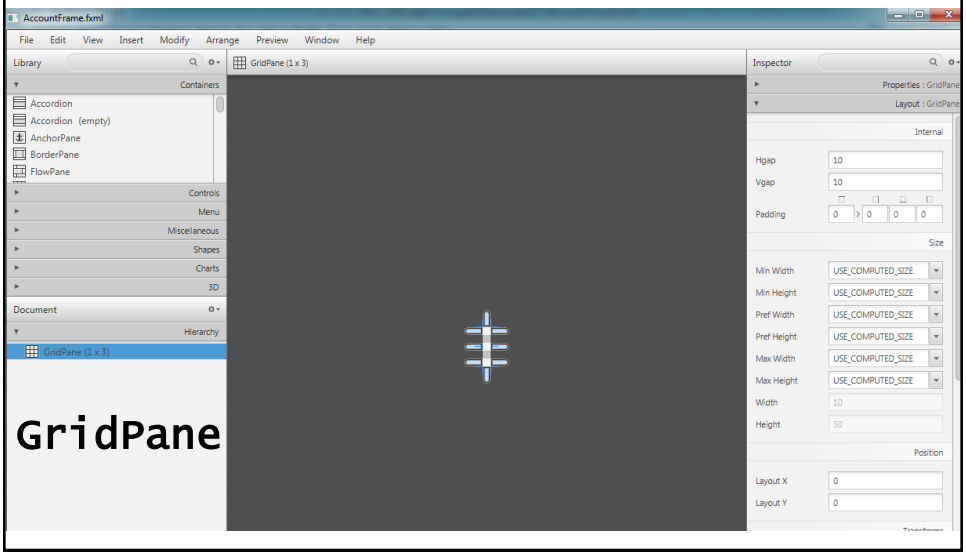
Bijlage

JavaFX

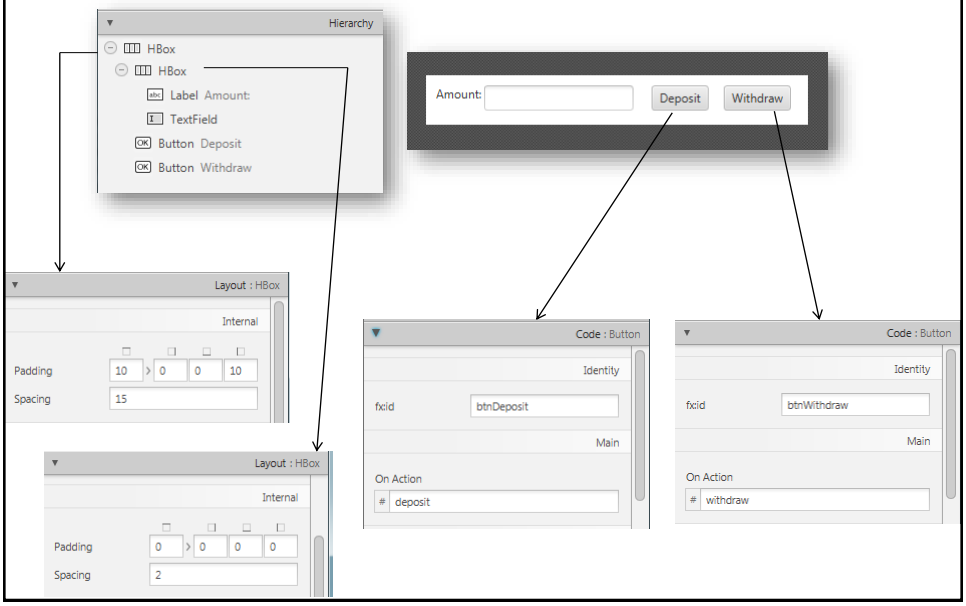
Scene Builder

HoGent

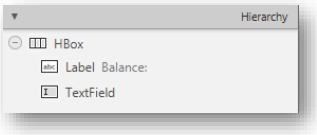
AccountFrame



AccountControllerPanel

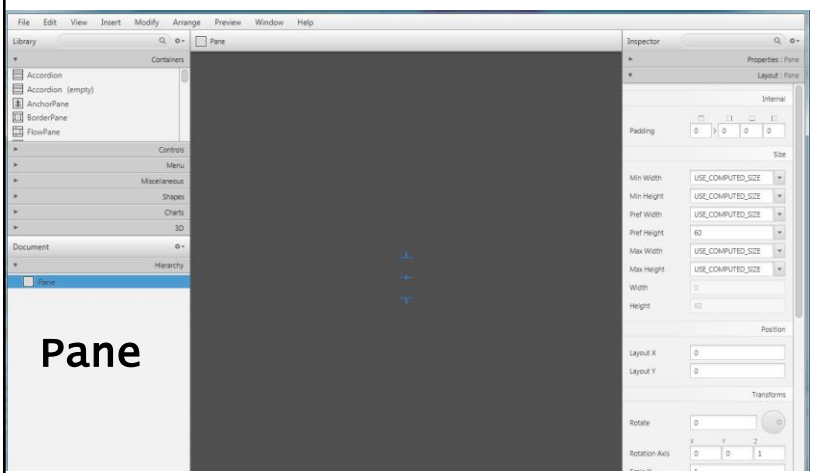


AccountTextViewPanel



HoGent

AccountBarGraphViewPanel



HoGent